CCU-C504 #1 Block Commutation Mode

Revolution Speed Measuring in Block Commutation PWM Mode for brushless Synchronous AC Motors

The following application note concerns microcontrollers of the C500 family with a CCU Peripheral Unit on Chip which is similar to:

SAx-C504-LM
SAx-C504-2RM

In block commutation mode, a synchronous generated defined incoming digital signal pattern of e.g. hall sensors, which are applied to the INT0-2 inputs, is sampled. Each transition at the INT0-2 inputs results in a change of the state of the PWM outputs according to a implemented block commutation table in the CCU unit without using CPU load for this operation.

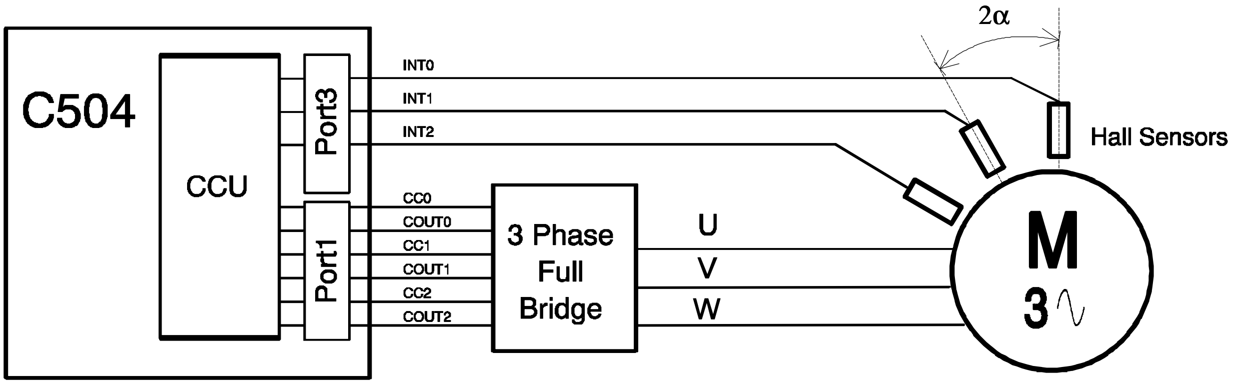

Figure1: Synchronous AC Motor in Block Commutation Mode

For monitoring the sensor input signal timing in block commutation mode, the signal transitions at INT0-2 external interrupt inputs can also generate an interrupt and a capture event at channel 0 of the CAPCOM unit in connection with compare timer 1. With this implemented feature, the revolution speed and acceleration or slow down phases of the synchronous motor can be easly recorded.

The rotor angle α of the motor between two generated eletrical hall sensor signals is dependent of the number of pairs of poles p of the machine. The formula is given by:

$$\alpha = \frac{60^o}{p}$$

In block commutation mode CAPCOM channel 0 is automatically configured in capture mode. Any signal transition at INT0-2 external inputs generates a capture pulse for CAPCOM channel 0 independently on the selected signal transition type rising or falling edge.The consequence is, that the 16-bit content at the former started compare timer 1 is captured at any signal transition at INT0-2 to the corresponding capture registers CCL0 and CCH0.

For calculating eg. the actual revolution speed of the synchronous motor, the corresponding external interrupt INT0-2 inputs now can be used for capturing the actual compare timer 1 values.

The difference between two directly followed 16-bit capture values than is a defined time base for the calculation of the actual revolution speed (regarding the number of pairs of poles of the machine). To get the difference from two directly followed compare timer 1 count values without calculation, practically in each external interrupt routine the actual capture value is saved and the compare timer 1 is restarted with count value 0000H in mode 0. Compare timer 1 overflows at period value FFFFH are not allowed and have to be recognized by the software to avoid wrong speed recording.

The revolution speed measure range is restricted by the following formula and parameters:

$$\mathbf{rps} = \frac{\mathbf{1}}{\mathbf{int\_no} \times \mathbf{period} \times \mathbf{edge} \times \mathbf{p}} [\mathbf{Hz}]$$

•	int_no	...	number of used external interrupt inputs (possible: 1 or 3)
•	rps	...	revolutions per second [Hz]
•	period	=	(resolution x countvalue)
	resolution	...	resolution of compare timer 1 dependent on the selected input clock prescaler ratio (fosc/2 - fosc/256)
	countvalue	...	0001H for minimum count value of compare timer 1 (... -> rps_max)
			FFFFH for maximum count value of compare timer 1 (... -> rps_min)
•	p	...	number of pairs of poles
•	edge	...	1 for rising **or** falling edge sensitive external interrupt input
		...	2 for rising **and** falling edge sensitive external interrupt input

The following table gives an examle of the range for the revolution speed measuring dependent of the fixed parameters p = 1, edge = 2 and int_no = 3. Compare timer 1 (COMPT1) input clock prescaler ratio is variable.

COMPT1 input clock	p [n]	edge [1 or 2]	int_no [1 or 3]	period_min (COMPT1) [s]	period_max COMPT1 [s]	rps_min [Hz]	rps_max (theoretical) [Hz]
fosc/2	1	2	3	$5x10^{-8}$	$3.28x10^{-3}$	50.86	$3.33x10^{8}$
fosc/4	1	2	3	$1x10^{-7}$	$6.55x10^{-3}$	25.43	$1.66x10^{7}$
fosc/8	1	2	3	$2x10^{-7}$	$13.11x10^{-3}$	12.71	$8.33x10^{6}$
fosc/16	1	2	3	$4x10^{-7}$	$26.21x10^{-3}$	6.35	$4.16x10^{6}$
fosc/32	1	2	3	$8x10^{-7}$	$52.43x10^{-3}$	3.18	$2.08x10^{6}$
fosc/64	1	2	3	$1.6x10^{-6}$	$104.86x10^{-3}$	1.59	$1.04x10^{6}$
fosc/128	1	2	3	$3.2x10^{-6}$	$209.72x10^{-3}$	0.79	$5.21x10^{5}$
fosc/256	1	2	3	$6.4x10^{-6}$	$419.43x10^{-3}$	0.40	$2.60x10^{4}$

Table 1: Calculated examples for revolution speed measuring

The shadowed column in the table with rps_max are the theoretical calculated maximum frequencies of a revolution measurement with COMPT1 count value 0001H which practical cannot be achieved because of the restricted performance of the microcontroller and the mechanical restrictions of the motor. The column rps_min represents the minimim resolution of the revolution speed measurement with COMPT1 count value FFFFH.
The optimal parameters have to be selected according to the individual application demands.

The following C51 source code listing example uses edge=2, int_no=1 and fosc/256 prescaler ratio (p is dependent of the used motor). This application example is intended as a moment record of the revolution speed. If there is a demand for recording the acceleration or slow down phases of the brushless AC motor, the example has to be extended by a revolution speed value array which eg. can be accessed to with pointers.

HL MCB AT 1

K. Scheibert

Appendix:

- 4 pages of C51 source and assembler listings

```
C51 COMPILER V5.02,  RPS                                                                          05/03/96  09:28:02  PAGE 1


DOS C51 COMPILER V5.02, COMPILATION OF MODULE RPS
OBJECT MODULE PLACED IN RPS.OBJ
COMPILER INVOKED BY: C:\C51\BIN\C51.EXE RPS.C CD SB PL(100) PW(127) ROM(LARGE) SMALL OT(6,SPEED)

stmt level    source

   1          /* ---------------------------------------------------------------------------------------
   2           file rps.c
   3
   4          Copyright 1996 Siemens AG HL MC AT 1, Klaus Scheibert, 03/04/96
   5
   6          Keil C51 Compiler V5.02
   7
   8          ************************SW for Application Note CCU-504  #1 *********************************
   9
  10          *** Revolution Speed Measuring in Block Commutation PWM Mode for brushless AC Motors ***
  11
  12          fixed parameters: edge = 2, int_no = 1 and prescaler ratio of compare timer 1 = fosc/256
  13
  14          ------------------------------------------------------------------------------------- */
  15
  16          #pragma DEBUG OBJECTEXTEND CODE      /* command lines directives */
  17
  18          #pragma NOLISTINCLUDE
  19
  20          #include <reg504.h>                  /* special function register declarations for C504 */
  21
  22          #include <intc504.h>                 /* interrupt number definition file */
  23
  24          #define FOREVER for(;;)              /* endless loop */
  25
  26
  27
  28
  29          /*********************** GLOBAL DECLARATION OF VARIABLES ********************************/
  30
  31          union { unsigned int capture_reg;
  32               unsigned char capture_byte[2];
  33                } capture_union;
  34
  35          #define capture_low  capture_union.capture_byte[1]
  36          #define capture_high capture_union.capture_byte[0]
  37          #define capture_value capture_union.capture_reg
  38
  39                                               /* with this union, a copie from the capture register
  40                                                  of the comparetimer 1 is located in data and
  41                                                  is available as unsigned int and char */
  42
  43
  44          unsigned int data difference_val;    /* a copie of the cature value is done to difference_val */
  45          unsigned char data compt1_ov_val;    /* value is used for test of compare timer 1 overflow */
  46
  47
  48          /************************************** PROCEDURES *********************************************/
  49
  50
  51          void comp_ini (void)    /* SET CAPCOM UNIT IN BLOCK COMMUTATION MODE */
  52          {
  53   1        CT1OFH  = 0x00;       /* set offset register value to reset values */
  54   1        CT1OFL  = 0x00;
  55   1        CCPH    = 0x0FF;      /* set period register value to maximum values */
  56   1        CCPL    = 0x0FF;      /* period = 6.4us - 419.43ms @ fosc=40MHz and prescalers set to fosc/256 */
  57   1        COINI  |= 0x0FF;      /* compare ini reg CCx=1, COUTx=1 in inactive state */
  58   1        CMSEL0 |= 0x33;       /* capture mode enabled */
  59   1        CMSEL1 |= 0x03;       /* in block commutation mode CAPCOM channel 0 is  */
  60   1                              /* automatically configured for capture mode and captured in CCL0/CCH0*/
  61   1        TRCON   = 0x00;       /* CTRAP function disabled at CCx/COUTx pins */
  62   1        BCON    = 0x05;       /* block commutation mode is selected with rotate right */
  63   1                              /* bit BCEN is set - block commutation mode enabled */
  64   1        CT1CON  = 0x0F;       /* start CT1 in mode 0, prescaler set to fosc/256
  65   1                                 (period 6.4us - 419.43ms) */
  66   1      }
  67
  68
  69          void int_ini (void)    /* interrupt initialization */
  70          {
  71   1        IT0    = 1;
  72   1        EX0    = 1;          /* enable external interupt0 in at P3.2 */
  73   1        ITCON  = 0x03;       /* rising/falling edge-triggered mode at interrupt 0 */
  74   1
  75   1        CCIE  |= 0x80;       /* set ECTP bit for enabling period interrupt */
  76   1        IEN1  |= 0x20;       /* enable compare timer 1 interrupt */
  77   1
  78   1        EA     = 1;          /* enable all interrupts */
  79   1      }
  80
  81
  82          void rps_sub (void)    /* subroutine for saving the captured compare timer value to
  83                                    difference_val */
  84
  85          {
  86   1        if (compt1_ov_val > 0)           /* test error condition before saving capture value */
  87   1        {
  88   2          difference_val = 0x00;         /* if an comt1 overflow occured set difference_val to 0x00 */
  89   2          compt1_ov_val  = 0x00;         /* reset compare timer 1 error condition */
  90   2        }
  91   1        else
C51 COMPILER V5.02,  RPS                                                        05/03/96  09:28:02  PAGE 2

1         {
2          difference_val = capture_value;          /* save capture_value to difference_val */
2         }
1       }


        void extintx_sub (void)    /* subroutine for external interrupt0 */
        {
1         capture_high = CCH0;
1         capture_low = CCL0;      /* load int cature_value */
1         CT1CON = 0x17;           /* stop and reset compare timer 1; ct1clock = fosc/256 */
1         CT1CON = 0x0F;           /* start comparetimer with 0000H*/
1                                  /* 6 instruction cycles measurement error = 1.8 us @ fosc 40 MHz) */
1       }

        /********************************************** INTERRUPTS ***********************************/


        void comptim1 (void) interrupt COMT1 using 1

                                        /* #define  COMT1     13   // (73H) Compare Timer 1 interrupt */

        {
1          CCIR &= 0x7F;                     /* clear interrupt flag CT1FP */
1          compt1_ov_val = compt1_ov_val+1;  /* set error condition */
1       }


        void extint0 (void) interrupt EXTI0 using 1  /* EXTI0 = 0 */

                                        /* #define  EXTI0      0   // (03H) external interrupt 0 */


        {
1         extintx_sub ();                   /* call external interrupt 0 subroutine */
1         rps_sub ();                       /* call subroutine for calculation of rpm */
1       }


        /************************************ MAIN PROGRAM *****************************************/
        void main (void)
        {
1
1         comp_ini ();      /* call compare timer initialisation procedure */
1                           /* set block commutation mode with rotate right */
1         int_ini ();       /* subroutine for enable interrupt0 */
1
1         FOREVER           /* endless loop */
1         {
2         ;
2         }
1       }
```

```
C51 COMPILER V5.02,  RPS                                                     05/03/96  09:28:02  PAGE 3

ASSEMBLY LISTING OF GENERATED OBJECT CODE


             ; FUNCTION comp_ini (BEGIN)
                                           ; SOURCE LINE # 51
                                           ; SOURCE LINE # 52
                                           ; SOURCE LINE # 53
0000 E4            CLR     A
0001 F5E7          MOV     CT1OFH,A
                                           ; SOURCE LINE # 54
0003 F5E6          MOV     CT1OFL,A
                                           ; SOURCE LINE # 55
0005 75DFFF        MOV     CCPH,#0FFH
                                           ; SOURCE LINE # 56
0008 75DEFF        MOV     CCPL,#0FFH
                                           ; SOURCE LINE # 57
000B 75E2FF        MOV     COINI,#0FFH
                                           ; SOURCE LINE # 58
000E 43E333        ORL     CMSEL0,#033H
                                           ; SOURCE LINE # 59
0011 43E403        ORL     CMSEL1,#03H
                                           ; SOURCE LINE # 61
0014 F5CF          MOV     TRCON,A
                                           ; SOURCE LINE # 62
0016 75D705        MOV     BCON,#05H
                                           ; SOURCE LINE # 64
0019 75E10F        MOV     CT1CON,#0FH
                                           ; SOURCE LINE # 66
001C 22            RET
             ; FUNCTION comp_ini (END)

             ; FUNCTION int_ini (BEGIN)
                                           ; SOURCE LINE # 69
                                           ; SOURCE LINE # 70
                                           ; SOURCE LINE # 71
0000 D288          SETB    IT0
                                           ; SOURCE LINE # 72
0002 D2A8          SETB    EX0
                                           ; SOURCE LINE # 73
0004 759A03        MOV     ITCON,#03H
                                           ; SOURCE LINE # 75
0007 43D680        ORL     CCIE,#080H
                                           ; SOURCE LINE # 76
000A 43A920        ORL     IEN1,#020H
                                           ; SOURCE LINE # 78
000D D2AF          SETB    EA
                                           ; SOURCE LINE # 79
000F 22            RET
             ; FUNCTION int_ini (END)

             ; FUNCTION rps_sub (BEGIN)
                                           ; SOURCE LINE # 82
                                           ; SOURCE LINE # 85
                                           ; SOURCE LINE # 86
0000 E500    R     MOV     A,compt1_ov_val
0002 D3            SETB    C
0003 9400          SUBB    A,#00H
0005 4008          JC      ?C0003
                                           ; SOURCE LINE # 87
                                           ; SOURCE LINE # 88
0007 E4            CLR     A
0008 F500    R     MOV     difference_val,A
000A F500    R     MOV     difference_val+01H,A
                                           ; SOURCE LINE # 89
000C F500    R     MOV     compt1_ov_val,A
                                           ; SOURCE LINE # 90
000E 22            RET
000F         ?C0003:
                                           ; SOURCE LINE # 92
                                           ; SOURCE LINE # 93
000F 850000  R     MOV     difference_val,capture_union
0012 850000  R     MOV     difference_val+01H,capture_union+01H
                                           ; SOURCE LINE # 94
                                           ; SOURCE LINE # 95
0015         ?C0005:
0015 22            RET
             ; FUNCTION rps_sub (END)

             ; FUNCTION extintx_sub (BEGIN)
                                           ; SOURCE LINE # 98
                                           ; SOURCE LINE # 99
                                           ; SOURCE LINE # 100
0000 85C300  R     MOV     capture_union,CCH0
                                           ; SOURCE LINE # 101
0003 85C200  R     MOV     capture_union+01H,CCL0
                                           ; SOURCE LINE # 102
0006 75E117        MOV     CT1CON,#017H
                                           ; SOURCE LINE # 103
0009 75E10F        MOV     CT1CON,#0FH
                                           ; SOURCE LINE # 105
000C 22            RET
             ; FUNCTION extintx_sub (END)

             ; FUNCTION comptim1 (BEGIN)
                                           ; SOURCE LINE # 110
                                           ; SOURCE LINE # 115
0000 53E57F        ANL     CCIR,#07FH
                                           ; SOURCE LINE # 116
```

```
C51 COMPILER V5.02,  RPS                                                    05/03/96  09:28:02  PAGE 4

0003 0500    R     INC     compt1_ov_val
                                              ; SOURCE LINE # 117
0005 32            RETI
             ; FUNCTION comptim1 (END)

             ; FUNCTION extint0 (BEGIN)
0000 C0E0          PUSH    ACC
0002 C0D0          PUSH    PSW
                                              ; SOURCE LINE # 120
                                              ; SOURCE LINE # 126
0004 120000  R     LCALL   extintx_sub
                                              ; SOURCE LINE # 127
0007 120000  R     LCALL   rps_sub
                                              ; SOURCE LINE # 128
000A D0D0          POP     PSW
000C D0E0          POP     ACC
000E 32            RETI
             ; FUNCTION extint0 (END)

             ; FUNCTION main (BEGIN)
                                              ; SOURCE LINE # 132
                                              ; SOURCE LINE # 133
                                              ; SOURCE LINE # 135
0000 120000  R     LCALL   comp_ini
                                              ; SOURCE LINE # 137
0003 120000  R     LCALL   int_ini
                                              ; SOURCE LINE # 139
0006         ?C0009:
                                              ; SOURCE LINE # 140
                                              ; SOURCE LINE # 142
0006 80FE          SJMP    ?C0009
                                              ; SOURCE LINE # 143
0008 22            RET
             ; FUNCTION main (END)
```